Numerical Analysis (2/7)
Interpolation - approximation

University of Luxembourg - 2024

Master in Mathematics

Hadrien Beriot

Outline

1. Interpolation overview

2. Polynomial interpolation

3. Approximation – data fitting

1. Interpolation overview

Discrete Data

t	1.0	2.0	3.0	4.0	5.0	6.0
y	1.9	2.7	4.8	5.3	7.1	2.4

- There are many different things we might want to do with such data…
  - Plot data on a graph and draw a curve through the data points
  - Infer data values between the points or predict values beyond the available data
  - Determine important parameters (e.g. birth/death rates) if the data represents a physical phenomenon
  - Approximate the derivative or integral, or quickly evaluate the function, if the data represents an underlying function
  - …
- **Objective**: represent discrete data in terms of relatively **simple functions** for easy manipulation

The Importance of Interpolation

- Traditionally, interpolation was used to compute approximate values of mathematical functions between tabulated values
- Advent of computers/calculators has reduced this application, but interpolation remains a cornerstone of numerical analysis
- But interpolation is **more than just a data manipulation technique**
  - Discrete → Continuous:
    - representing discrete data as a function (vs. a table) is key to bridging finite- and infinite-dimensional problems

SIEMENS

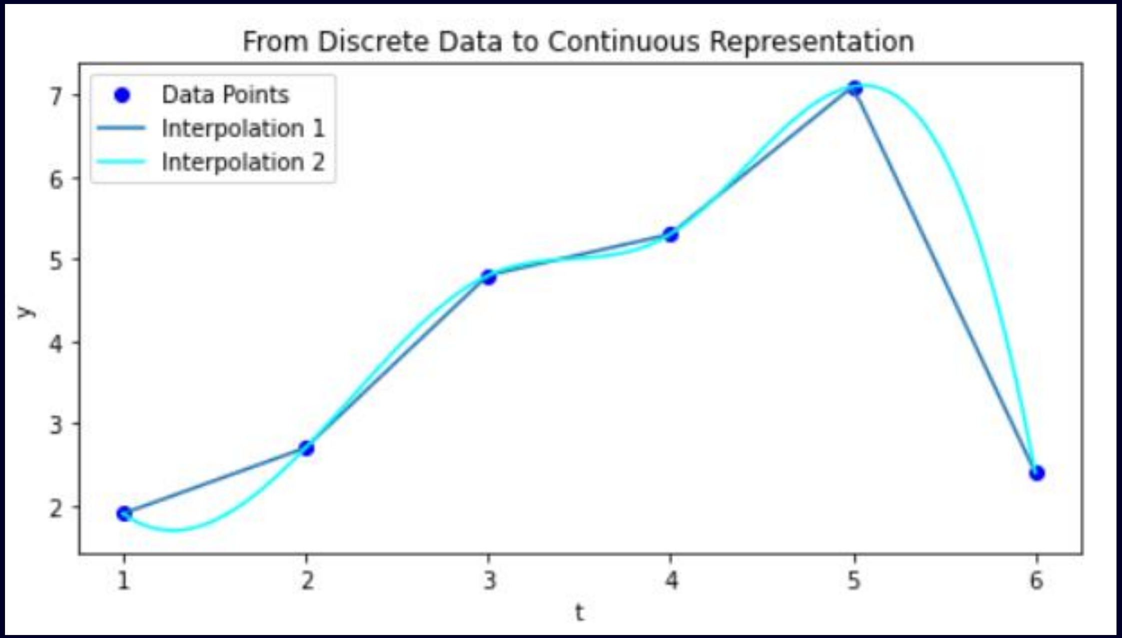
The Importance of Interpolation

Discrete

interpolation

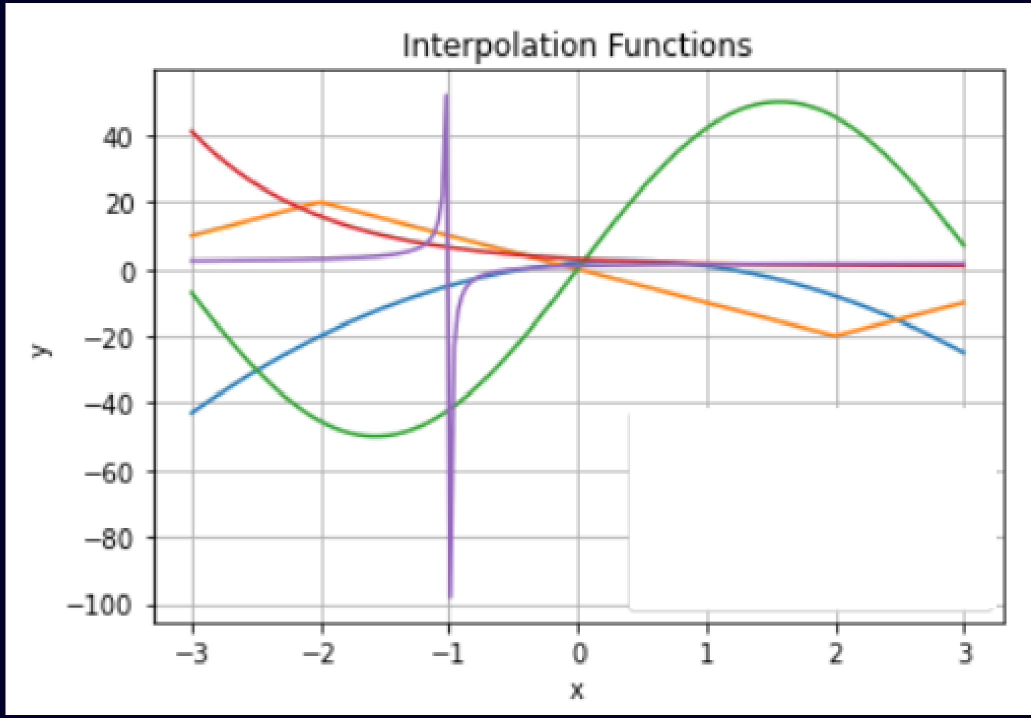
Continuous

t	1.0	2.0	3.0	4.0	5.0	6.0
y	1.9	2.7	4.8	5.3	7.1	2.4



Type of functions used in interpolation

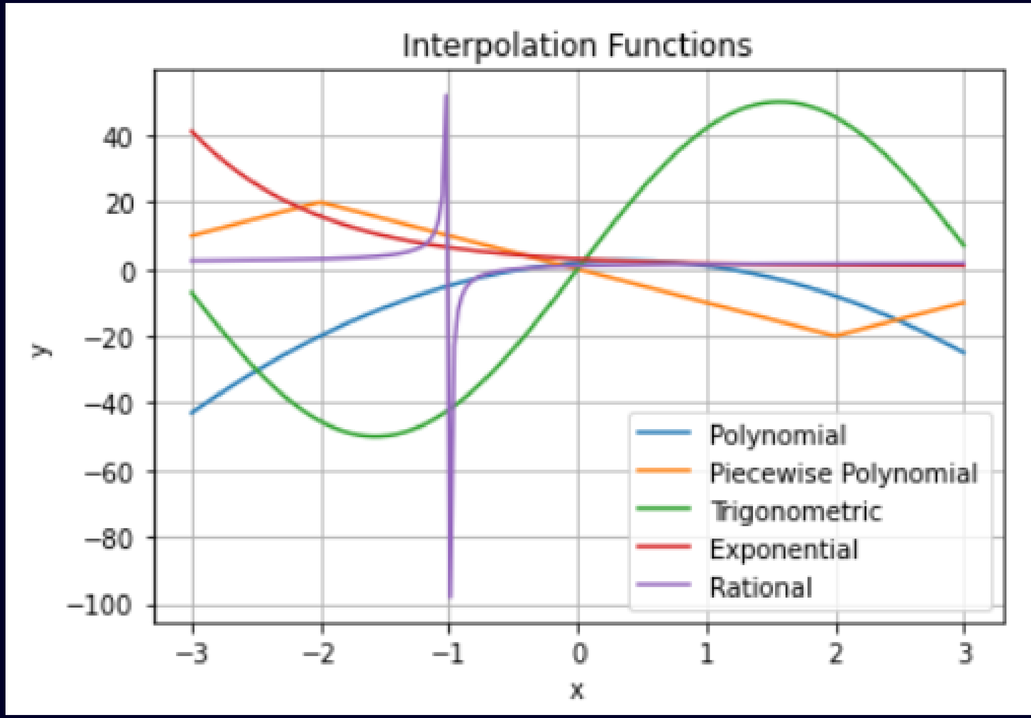
- Exercise:
  - Can you recognize some of the function types used here?



SIEMENS

Type of functions used in interpolation

- **Polynomials**: $P(x) = a_0 + a_1x + a_2x^2 + \dots + a_nx^n$
- **Piecewise polynomials**: $P(x) = \{ P_1(x), for\ x \in [a,b)\ P_2(x), for\ x \in [b,c)\ \dots \}$
- **Trigonometric** : e.g. $P(x) = \sin(\omega x)$
- **Exponential** : e.g. $P(x) = a_0 + a_1 e^{b_x}$
- **Rational** : $P(x) = \frac{a^0 + a^1x + a^2x^2 + \dots + a_nx^n}{b_0 + b_1x + b_2x^2 + \dots + b_mx^m}$



SIEMENS

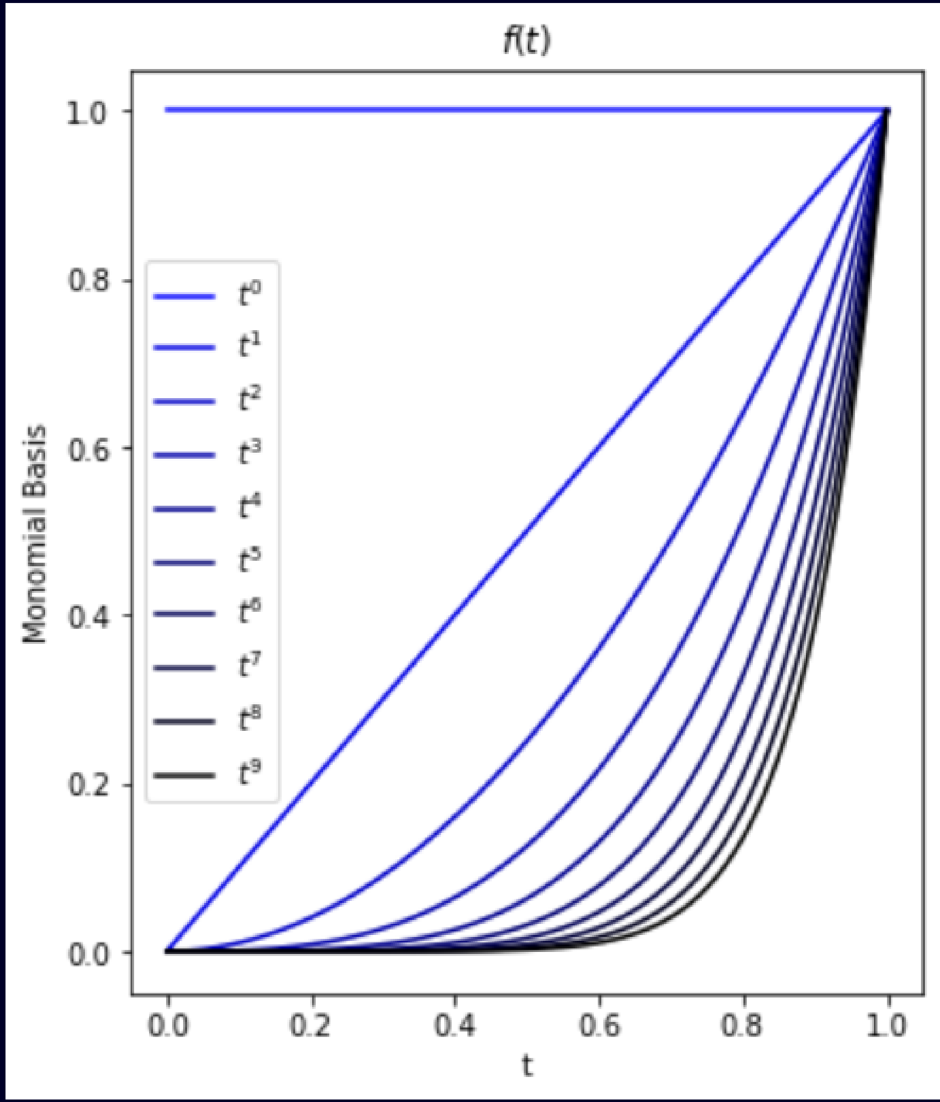
2. Polynomial Interpolation

Monomial basis

- To interpolate n data points: $k = n - 1$
- Monomials are the most natural basis for P_{n-1}
  - $p_{n-1}(t) = x_1 + x_2 t + \cdots + x_n t^{n-1}$

$$\boldsymbol{A}\boldsymbol{x} = \begin{bmatrix} 1 & t_1 & \cdots & t_1^{n-1} \\ 1 & t_2 & \cdots & t_2^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & t_n & \cdots & t_n^{n-1} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \boldsymbol{y}.$$

- A matrix of this form is called a Vandermonde matrix
- This system gives the coefficients for interpolating the data points (t_i, y_i)
- This matrix is often ill-conditioned for high-degree polynomials. Do you know why?



Monomial basis:

To illustrate polynomial interpolation using the monomial basis, we will determine the polynomial of **degree two (quadratic)** interpolating the three data points

$$(-2, -27), (0, -1), (1, 0)$$

Assemble the Vandermonde Matrix manually and solve it using Gaussian elimination

Lagrange basis

- Consider $n + 1$ distinct points $x_0, x_1, \dots x_n$ and $y_0, y_1, \dots y_n$ on the interval $[a, b]$.
- There exists one and only one polynomial P_n of degree less or equal to n satisfying

$$P_n(x_i) = y_i, \forall i = 0, 1, \dots, n$$

- which writes

$$P_n(x) = \sum_{i=0}^{n} y_i L_i(x) \text{ with } \mathrm{L_i(x)} = \prod_{k=0, k \neq i}^{n} \frac{(x - x_k)}{(x_i - x_k)}$$

- This polynomial Pn is called Lagrange interpolation polynomial at the points $x_0, x_1, \dots x_n$.
- The polynomials $L_i\ (x)$ are the Lagrange basis functions associated to the points xi:
  - At the points $x_0, x_1, \dots x_n$, only one basis function is “active” and equal to 1 (Vandermonde is identity)
  - making Lagrange interpolation straightforward and intuitive

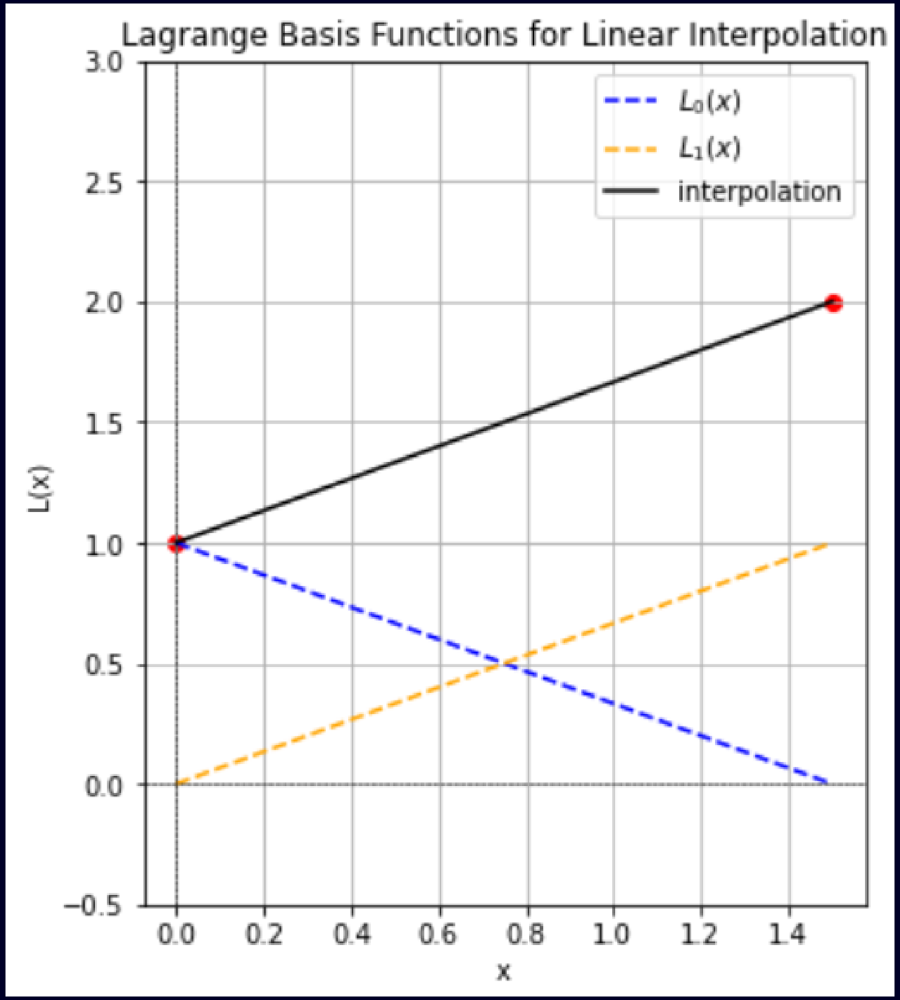
Lagrange Interpolation

- With two points $n = 1$ (linear)

$$P_1(x) = y_0 \frac{x - x1}{x_0 - x_1} + y_1 \frac{x - x0}{x_1 - x_0}$$

- With three points $n = 2$ (quadratic)

→ Exercise #1



SIEMENS

How good is polynomial interpolation?

- For a continuous function $f(x)$, what is the approximation error when we replace it by $P_n(x)$?

$$E_n(x) = f(x) - P_n(x), \qquad x \in [a,b]$$

- Theoretical result: if f has $n + 1$ bounded derivatives:

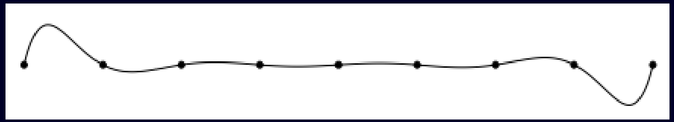
$$E_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}\,\pi_n(x), \qquad \xi \in [a,b]$$

- Where $\pi_n(x) = \prod_{i=0}^{n}(x - x_i)$.
- If x is node than E_n=0, which is expected ☺
- The error depends on two terms, smoothness of function and number of placement of nodes. More precisely
  1. The $(n+1)^{th}$ derivative of f
  2. the maximum of the function π_n, which only depends on the choice of x_i

SIEMENS

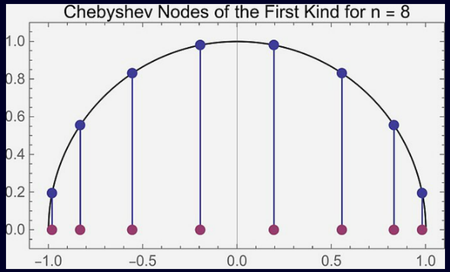
Choice of interpolation points

- We can try to minimize $\max|\pi_n(x)|$ by a better choice of the points x_i
- Choosing equally distributed points is far from being optimal! $\pi_8(x) = \prod_{i=0}^{8}(x - x_i)$



- In some occasions, it is unstable, error can grow with polynomial order (Runge phenomenon)
- The optimal choice is obtained by the Chebyshev points

$$x_i = \frac{a+b}{2} + \cos\left(\frac{(2i+1)\pi}{2n+2}\right)\frac{b-a}{2}$$



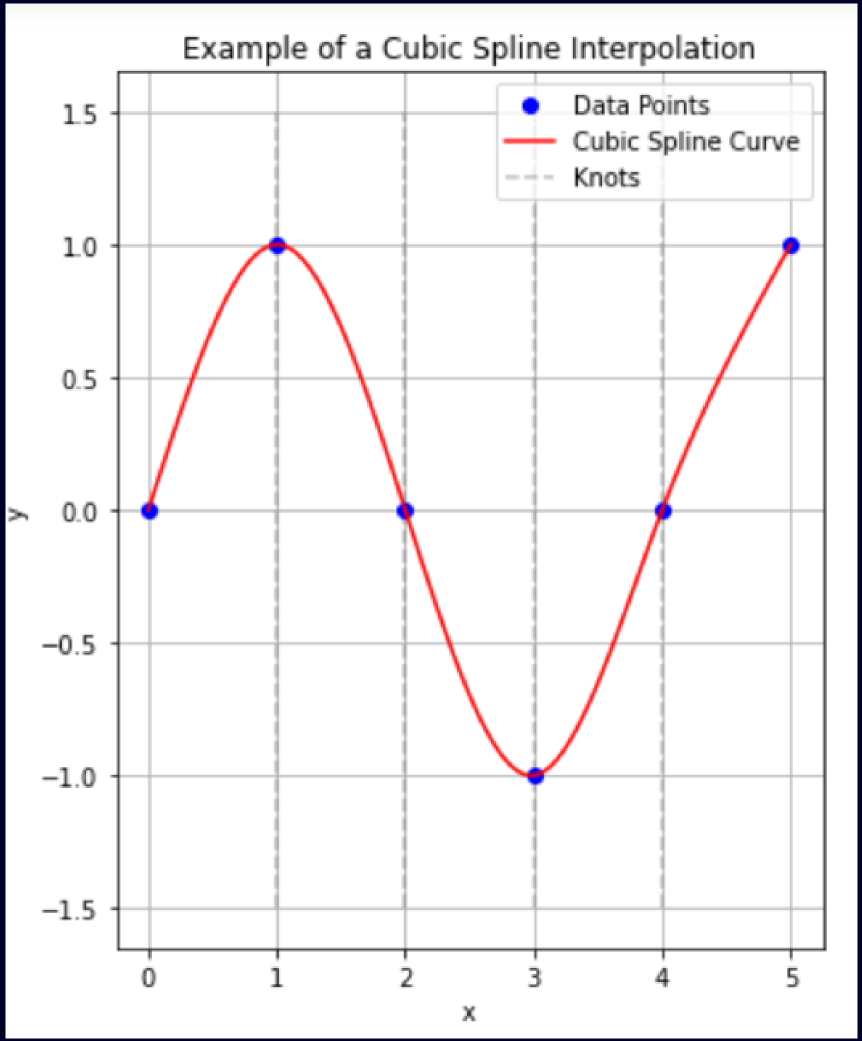
- → Exercise #2
- Another way to avoid Runge phenomena is by **piecewise interpolation (linear, quadratic, spline)**

SIEMENS

Cubic splines

Degree 3, twice continuously differentiable.

- **Constraints for Cubic Splines**
  - **Interpolation & First Derivative Continuity**: $3n-4$ constraints.
  - **Continuous Second Derivative**: Adds $n-2$ constraints.
  - **Remaining**: 2 free parameters. (can you explain why?)




```
# Create a cubic spline interpolation of the data points
cs = CubicSpline(x, y, bc_type='natural')
```

Parameter Setting Methods

1. **First Derivative:** estimated at Endpoints
2. **Natural**: Second derivative = 0 at endpoints.
3. **"Not-a-Knot"**: Matches segments at internal points.
4. **Periodic**: Matches 1st and 2nd derivatives at endpoints.

→ Exercise #3

SIEMENS

Orthogonal polynomials

- A sequence of orthogonal polynomials is such that $\phi_0(x), \phi_1(x), \cdots$ is such that

ϕ_i is of degree i

$$< \phi_i, \phi_j > = 0, \quad \text{if i} \neq j$$

- **Legendre polynomials** $L_n(x)$ are a set of orthogonal polynomials defined on the interval $[-1{,}1]$by

$$L_0 = 1, L_1 = x,$$

$$(n + 1)\, L_{n+1}(x) = (2n + 1)\, x\, L_n(x) - n\, L_{n-1}(x).$$

- They verify

$$< \phi_i, \phi_j >_{L^2} = \int_{-1}^{1} \phi_i(\xi)\phi_j(\xi)d\xi = 0$$

- Orthogonal polynomials often lead to well conditioned problems and play an important role in numerical analysis (Gaussian quadrature, approximation, solving differential equations)

SIEMENS

Orthogonal polynomials

$$< \phi_i, \phi_j >_{L^2} = \int_{-1}^{1} \phi_i(\xi)\phi_j(\xi)d\xi = 0$$

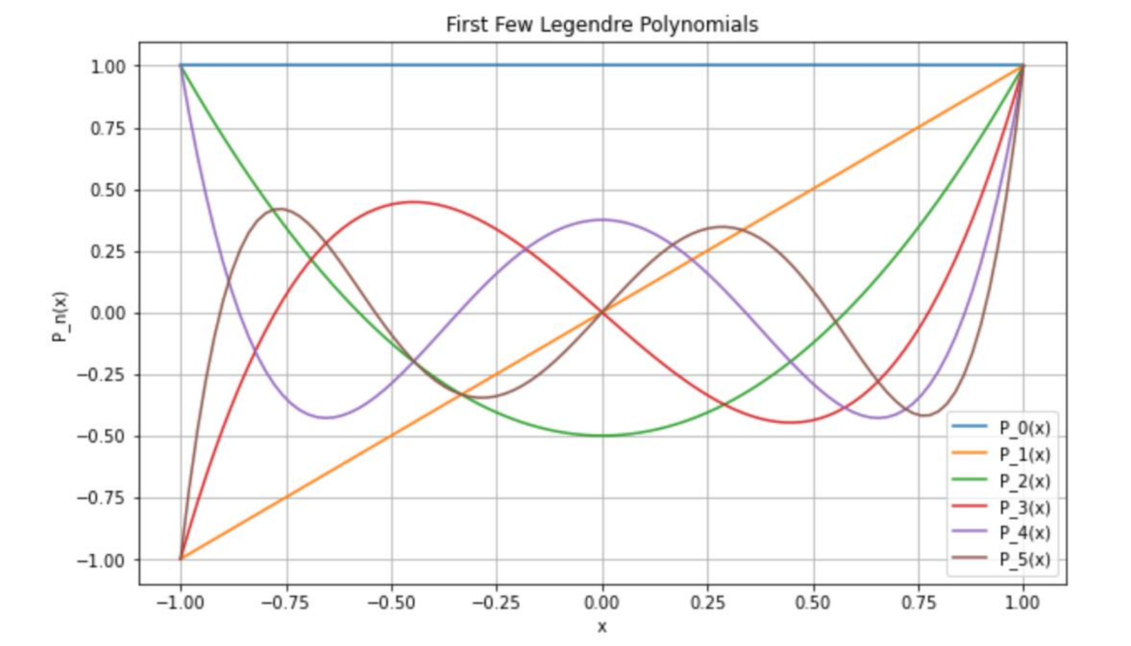
```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.special import legendre

# Define range of x values
x = np.linspace(-1, 1, 100)

# Number of polynomials to display
num_polynomials = 6

# Generate and plot each polynomial
plt.figure(figsize=(10, 6))
for n in range(num_polynomials):
    P_n = legendre(n)  # Get the nth Legendre polynomial
    plt.plot(x, P_n(x), label=f'P_{n}(x)')

plt.title('First Few Legendre Polynomials')
plt.xlabel('x')
plt.ylabel('P_n(x)')
plt.legend()
plt.grid(True)
plt.show()
```

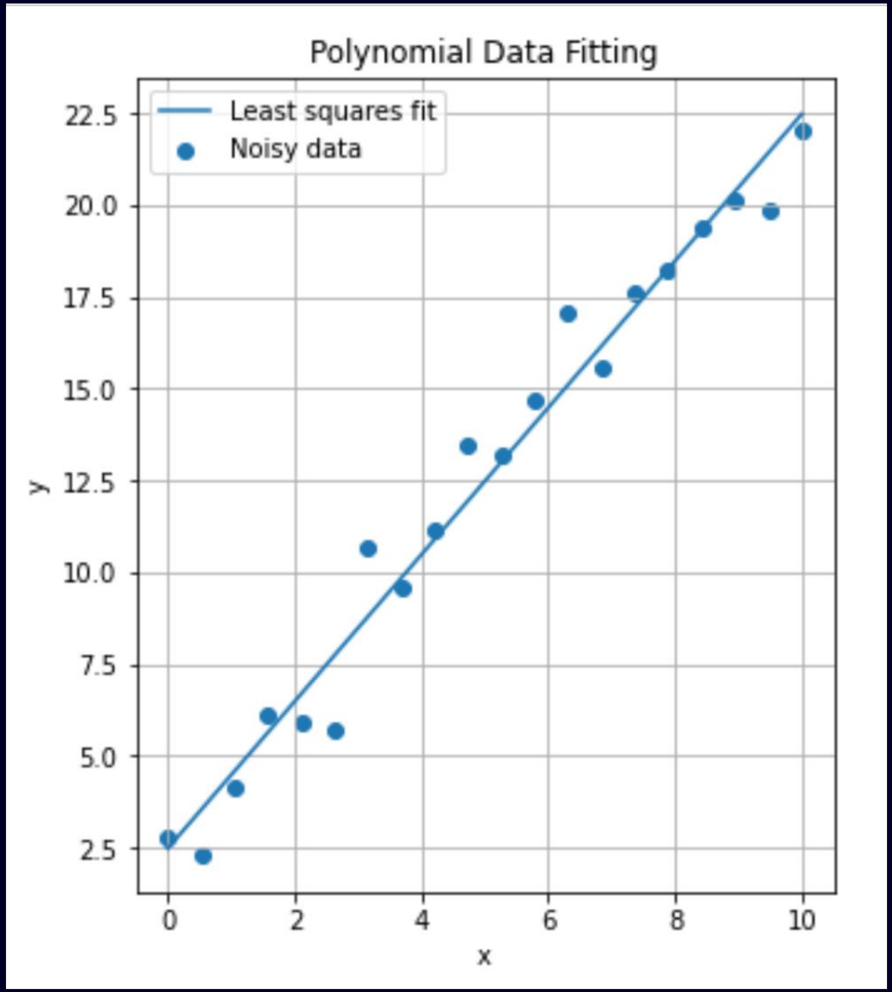


SIEMENS

3. Approximation – data fitting

Polynomial Interpolation vs approximation

- **Interpolation** = Exact Data Fit
- In some cases (noisy or too much data), exact data fit not always appropriate
- If data has experimental/significant errors, it's preferable **to smooth out noise**
  - We do not need P to pass through x_i , but only capture the main trend of f
  - we can try to fit the data to a polynomial P of lower degree n, we talk about **discrete least squares.**



Data fitting

t	1.0	2.0	3.0	4.0	5.0	6.0
y	1.9	2.7	4.8	5.3	7.1	2.4

- Given data points $(t_i, y_i), i = 1, \ldots, m$, we wish to find the n-vector α of parameters that gives the "best fit", where by best fit we mean in the least squares sense

$$\min_{\alpha} \sum_{i=1}^{m} \left(y_i - f(t_i, \alpha)\right)^2$$

- A data fitting is *linear* if the function f is linear in the components of the parameter vector $\boldsymbol{\alpha}$, which means that f is a linear combination

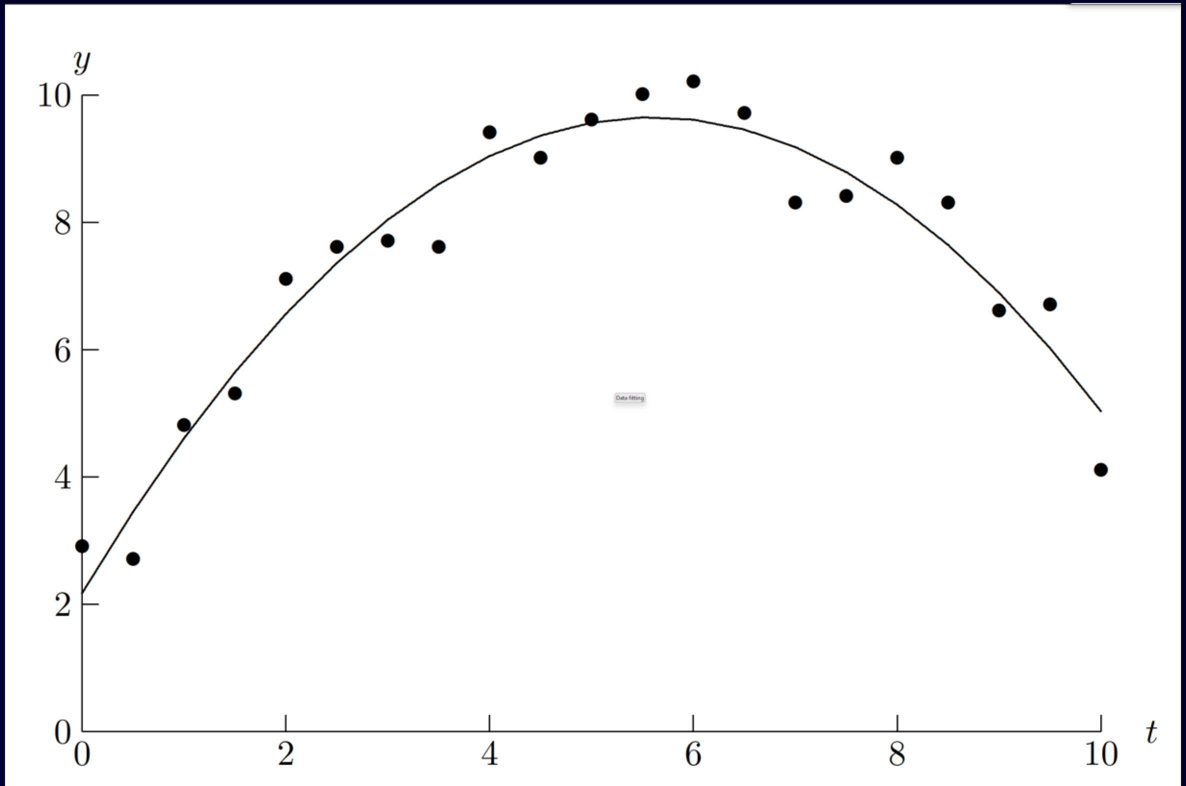
$$f(x,t) = \alpha_1 \phi_1(t) + \alpha_2 \phi_2(t) + \cdots + \alpha_n \phi_n(t)$$

- of functions ϕ_j that depend only on t. In least squares fitting, the algorithm essentially "finds" the best curve by minimizing the sum of squared errors between the data points and the fitted line

Linear data fitting - or linear regression

- For example, polynomial fitting, with $f(t, \alpha) = \alpha_1 + \alpha_2 t + \alpha_3 t^2$ is a linear data fitting problem, although it is obviously quadratic in t!

the "linear" part refers to the linearity in the α coefficients



Least squares fit of quadratic polynomial to given data.

SIEMENS

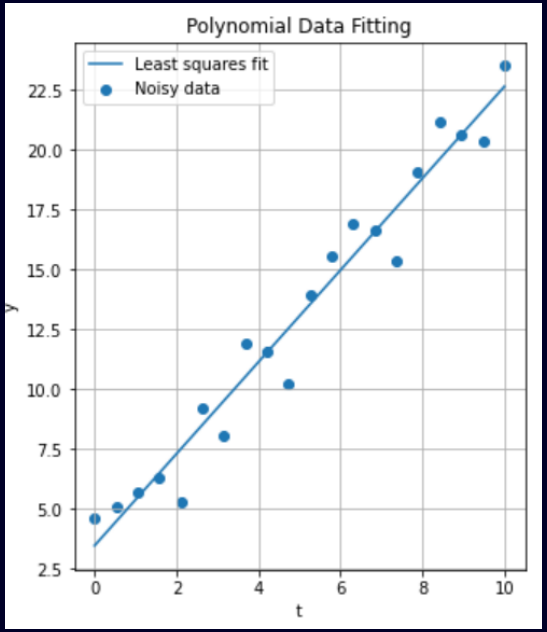
Data fitting - example

t	1.0	2.0	3.0	4.0	5.0	6.0
y	1.9	2.7	4.8	5.3	7.1	2.4

- Let us do some data fitting on this set of points using least squares
- At first sight it looks like a linear function will do a good job
- So we need to find α_1 and α_2 in the function $f(t, \alpha) = \alpha_1 + \alpha_2 t$

$$\epsilon = \sum_{i=1}^{6} \left(y_i - f(t_i, \boldsymbol{\alpha})\right)^2$$

- To minimize this function we can get $\frac{\partial \epsilon}{\partial \alpha_1} = 0$ and $\frac{\partial \epsilon}{\partial \alpha_2} = 0$
- $\frac{\partial \epsilon}{\partial \alpha_1} = -2 \sum_{i=1}^{6} \left(y_i - f(t_i, \boldsymbol{\alpha})\right) = 0$, and $\cdots$
- Which is complex…



SIEMENS

Data fitting - example

t	1.0	2.0	3.0	4.0	5.0	6.0
y	1.9	2.7	4.8	5.3	7.1	2.4

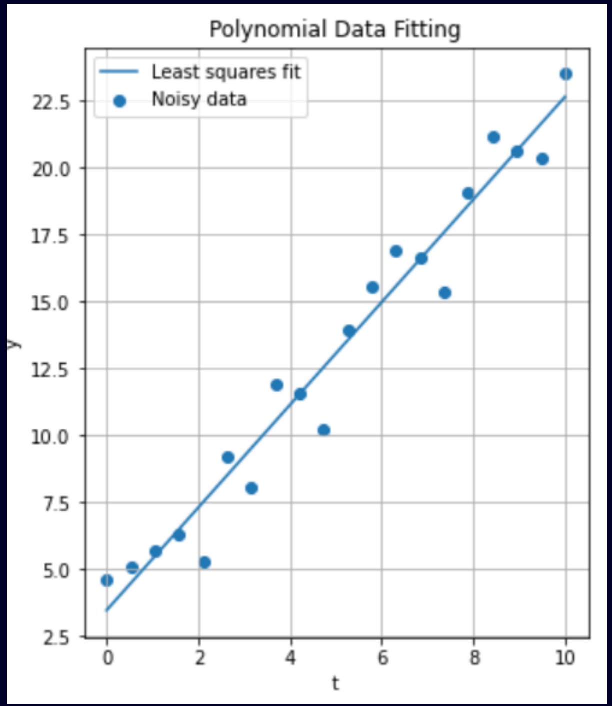
- Instead, it is easier to construct the Vandermonde matrix

$$\underbrace{\begin{bmatrix} 1 & t_1 \\ \vdots & \vdots \\ 1 & t_6 \end{bmatrix}}_{6\times 2} \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} \approx \begin{bmatrix} y_1 \\ \vdots \\ y_6 \end{bmatrix}$$

- The solution of this overdetermined system (we have more equations than unknowns)

$$\boldsymbol{Ax} \cong \boldsymbol{b}, \text{ with } m > n$$

- can be computed using linear algebra packages



$$f(t,\alpha) = \alpha_1 + \alpha_2 t$$

SIEMENS

Normal Equations

$$\underbrace{\begin{bmatrix} 1 & t_1 \\ \vdots & \vdots \\ 1 & t_6 \end{bmatrix}}_{6 \times 2} \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} = \begin{bmatrix} y_1 \\ \vdots \\ y_6 \end{bmatrix}$$

- We wish to minimize the squared Euclidean norm of the residual vector $\boldsymbol{r} = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}$

$$\phi(\boldsymbol{x}) = \|\boldsymbol{r}\|_2^2 = \boldsymbol{r}^T\boldsymbol{r} = (\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x})^T(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}) = \boldsymbol{b}^T\boldsymbol{b} - 2\boldsymbol{x}^T\boldsymbol{A}^T\boldsymbol{b} + \boldsymbol{x}^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x}.$$

- A necessary condition for a minimum is that $\boldsymbol{x}$ be a critical point of Φ, where the gradient vector $\nabla\Phi(\boldsymbol{x})$ is zero

$$\mathbf{0} = \nabla\phi(\boldsymbol{x}) = 2\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x} - 2\boldsymbol{A}^T\boldsymbol{b},$$

- We finally get to

$$\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}^T\boldsymbol{b}.$$

- This is the system of normal equations, or cross matrix product of A, which is a standard linear system

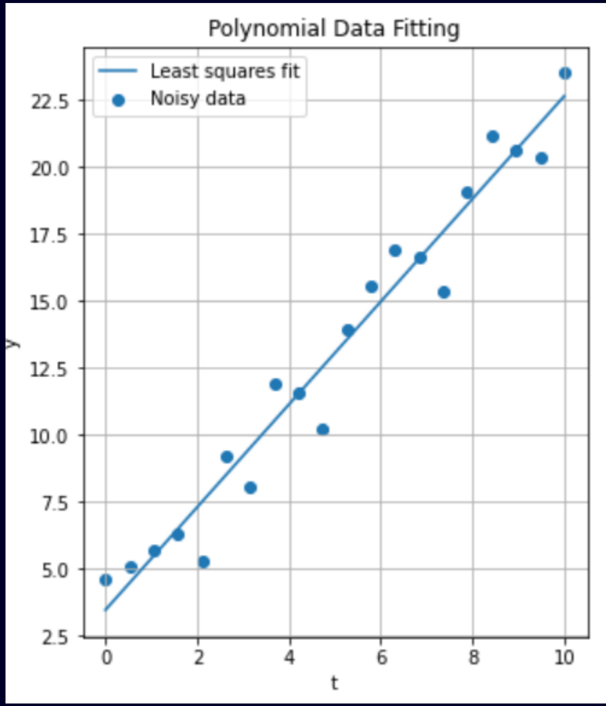
SIEMENS

Normal equation

t	1.0	2.0	3.0	4.0	5.0	6.0
y	1.9	2.7	4.8	5.3	7.1	2.4

$$\underbrace{\begin{bmatrix} 1 & \cdots & 1 \\ t_1 & \cdots & t_6 \end{bmatrix} \begin{bmatrix} 1 & t_1 \\ \vdots & \vdots \\ 1 & t_6 \end{bmatrix}}_{2\times 2} \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & \cdots & 1 \\ t_1 & \cdots & t_6 \end{bmatrix} \begin{bmatrix} y_1 \\ \vdots \\ y_6 \end{bmatrix}}_{2\times 1}$$

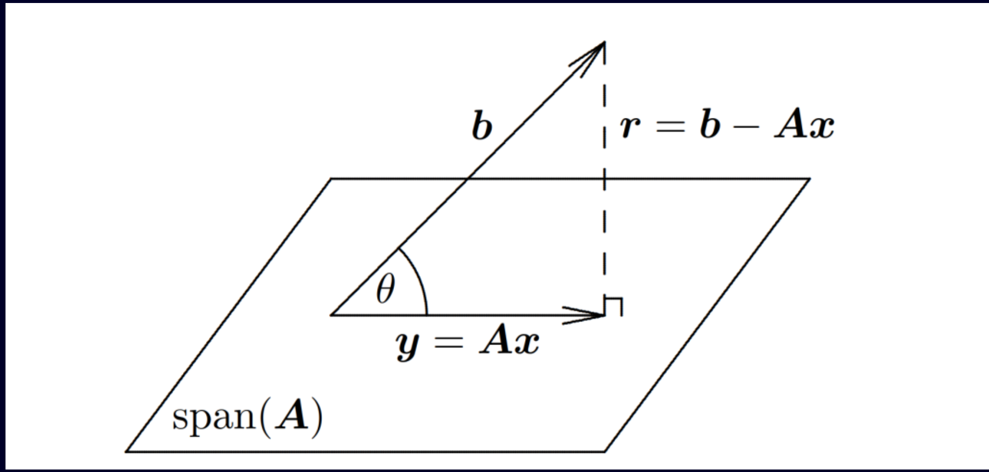
$$\boxed{A^T A x = A^T b.}$$



SIEMENS

Geometric interpretation

- The m-vector b generally does not lie in $span(\boldsymbol{A})$, a subspace of dimension at most n
- The vector **y = Ax ∈ span(A)** *closest* to $\boldsymbol{b}$ in the Euclidean norm occurs when the residual vector **r = b − Ax** is orthogonal to $span(\boldsymbol{A})$
- Residual vector $\boldsymbol{r} = \boldsymbol{b} - \boldsymbol{Ax}$ must be orthogonal to each column of $\boldsymbol{A}$, $\boldsymbol{A}^T \boldsymbol{r} = \boldsymbol{0}$



→ Exercise #4 and #5

1.6 Summary

Summary

- Polynomial interpolation :
- 1. Global polynomial interpolation (Lagrange, Legendre)
  - Error study: Runge phenomena, Chebyshev points
- 2. Local interpolation (piecewise linear, cubic splines) is more robust for non smooth functions
  - Higher degree local interpolation requires more data (new point, derivative(s), ...)
    - $\Rightarrow$ more unknowns to be solved
- Approximation theory :
  - Discrete least-squares is the main method used for data fitting
  - It requires solving the linear system of normal equations (although more advanced methods are often used)

